The 1996 Hub-4 Sphinx-3 System

P. Placeway, S. Chen, M. Eskenazi, U. Jain, V. Parikh, B. Raj,
M. Ravishankar, R. Rosenfeld, K. Seymore, M. Siegler, R. Stern, and E. Thayer

Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
pwp@cs.cmu.edu

ABSTRACT

This paper describes the CMU Sphinx-3 system, and the configuration we used for the 1996 DARPA (Hub-4) evaluation. The model structure, acoustic modeling, language modeling, lexical modeling, and system structure are summarized. We also discuss the experimental results obtained with this system on the most recent DARPA evaluation, and some subsequent results are also discussed.

Motivation

Past efforts on speech recognition have focused on clean, good quality speech in friendly environments, and DARPA evaluations in past years have followed this agenda. While one must walk before one can run, we have, as a community, developed our technology to the point where we can handle large vocabulary dictation well, and spontaneous speech fairly well.

The evaluations have tracked this, with the introduction last year of so-called "found" speech, recorded off the air from commercial broadcasts. For the 1996 Hub-4 evaluation we have continued in this vein, widening our horizons to include the additional difficulty of television news broadcasts, as well as multiple radio programs. This offered data containing many interesting effects, such as spontaneous speech, speech over wide- and narrow-band channels and other common degradations, and non-native speakers, as well as read, clean speech. It also confronted us with the difficulties of broadcast speech, including dynamic range compression and peak-limiting, speech-over-music and speech-over-speech acoustic production effects.

This evaluation also offered the chance to refine automatic segmentation techniques, and to compare the automatic segmentation to human-transcribed segmentation. The segmentation problem is an important one. If speech systems are to be employed in real-world settings such as these, they cannot depend on pre-marked acoustic and linguistic boundaries. Requiring human formatting of input data obviates many advantages of mechanical transcription; and there are simply too many broadcasts to consider this over the long-term.

So we were presented with interesting and useful new challenges in this evaluation. This paper presents an overview of our efforts to address these challenges.

System Overview

The Sphinx-3 system is our new, flexible, hidden Markov model-based speech recognition system. Its components can be configured at run-time along the spectrum of semi- to fully-continuous operation. We designed each component to do one thing well, have very few hard-wired limitations, and interface easily to other components. This has allowed easy and very flexible system design.

For this evaluation, we ran a senonically clustered [1], mixture-density, fully-continuous system. The acoustic model was trained on the Wall Street Journal corpus and adapted to a subset of the 1996 Hub-4 broadcast news training data in a supervised manner, then adapted toward the test data in an unsupervised manner. The language model was trained on the Broadcast News corpus. The lexicon was chosen from the most common words in this corpus, to be of a size that balances the trade-off between leaving words out-of-vocabulary, and introducing extra confusable words. Some common acronyms and phrases grouped into lexical units were also added as lexical units. Several types of filled pauses were also modeled explicitly.

The different models that the system used will be described next, followed by the overall recognition system architecture, and finally a discussion of results.

Acoustic Modeling

Model Structure and Base Training: The Sphinx-3 system is a Hidden Markov Model-based speech recognition system. The evaluation system used 6000 senonically-tied states [1], each consisting of a mixture of 16 diagonal-covariance Gaussian densities.

At the time, we did not feel that there was enough training data from the 1996 Hub-4 data collection effort to train our models adequately. We also felt that we had neither the personnel to prepare sufficient training data, nor the time to find an appropriate operating point for our system on the smaller amount of data that this effort would have yielded. Instead we chose to employ an adapted-training approach.

Two sets of acoustic models were trained: one for wide-bandwidth data, and one for narrow-bandwidth (*e.g.* telephone) data. Both of these models were mixed-gender. We had originally planned to use three different models for full-bandwidth data: male-only, female-only, and mixed-gender. However, tests on the development test set showed no advantage to the split-gender techniques on these data, so we used only the mixed-gender model for wide-bandwidth inputs.

The wide bandwidth acoustic model was constructed of two parts: context-dependent (CD) phonemes, and context-independent (CI) phonemes including noise phones. CD phonemes (triphones) were mapped into 6000 senones. These were first trained on the WSJ SI-284 training set, and then adapted in a supervised manner to the acoustics of the portions of the 1996 training data that were marked as either the clean, read (F0) or clean, spontaneous (F1) conditions. The CI portion of the model consisted of 52 CI phones and six noise phones, including *AH*, *UH*, and *UM*. These were trained on the F0 and F1 portions of the training data. Each phone or triphone was represented as a 5-state HMM; each tied state was a mixture of 16 densities.

The narrow bandwidth acoustic model was first trained on WSJ SI-321 with reduced bandwidth. This model was then adapted to the (1 hour) subset of the 1996 training data that we identified as speech over a telephone channel, using the classifier described below. The labeling of this subset was then hand-refined prior to training. This acoustic model was structured as 6000 senonically tied states mapped into triphones, plus 52 context-independent phones and 3 noise phones (including silence). Each phone or triphone was represented as a 5-state HMM; each tied state was a mixture of 16 densities.

The choice of which model to use for each segment was made in the preprocessing step of decoding, described below.

Acoustic Adaptation: Model parameters were adapted using a transformation of the mean vectors based on linear regression [4]. This adaptation was used in two ways. Initially, models trained on SI-284 were adapted in a supervised manner to some of the 1996 Hub-4 training data as described above. This supervised adaptation showed a 7% relative improvement on the clean, read (F0) portion of the 1996 Hub-4 development test set, though only a 2% improvement on the whole set. This was a somewhat smaller improvement than we had hoped for; we think that this was due to using only one regression class, rather than several [3].

During recognition, the above models were also adapted in an unsupervised manner using the initial pass recognition results. In the preprocessing step (below) sub-segments were clustered according to acoustic similarity. For each cluster, all the results from its constituent sub-segments were used jointly to adapt the acoustic models. The adapted models were used during the final recognition pass. This process gave us roughly 4–5% relative improvement on the evaluation conditions. We used only a single iteration of unsupervised adaptation; initial testing showed that multiple iterations did not further improve the recognition accuracy.

For a more detailed discussion of our current adaptation techniques, see [7].

Lexical Modeling

The recognition vocabulary consisted of the most frequent 51,000 words of the Broadcast News corpus, supplemented by some 200 multi-word phrases and some 150 acronyms. The vocabulary size was initially based on our experience with speech systems. A subsequent careful analysis of the trade-off between out-of-vocabulary rate vs. acoustic confusability showed that our choice was a very good one [10].

The phrases were selected by hand after observing recognition errors among the acoustic training data. Many of these errors were attributed to incorrect pronunciation rules. In these cases, pronunciations were manually chosen from a set of possibilities implied by the data. This technique was also used to select alternative pronunciations for words in spontaneous speech. The acronyms chosen were the most frequent ones in the Broadcast News corpus, and accounted for more than 90% of the acronym tokens in that corpus [10].

Language Modeling

Two language models were used in our system: one to guide the decoder in the actual recognition passes, and a different, larger one for rescoring the N-best output hypotheses.

The decoder used a Katz-smoothed trigram language model. This is a fairly standard language model, much like those which have been used in the DARPA speech recognition community for the past several years. As a space optimization, singleton trigrams and bigrams were excluded from this model. As a new feature, this language model incorporated cross-boundary trigrams.

As far as language modeling is concerned, there are two linguistic problems introduced by segmenting data. One is that single linguistic utterances can be sliced into several pieces. Contrawise, several linguistic utterances can be packed together into the same acoustic segment. In order to address the latter problem in this system, we included extra trigrams in our language model training to simulate unmarked utterance-boundaries [10].

In the last recognition pass we generated a set of N-best hypotheses in order to apply rescoring [9]. This allowed us to do two interesting things. The first was to optimize the language model weight and insertion penalty for the purpose of rescoring automatically, using Powell's algorithm [8]. While this is a simple technique, it would be difficult to overstate its usefulness. The weights for the language score as well as the word insertion penalty were optimized on development test data.

The second was to use a different, physically larger language model for rescoring than could be used inside the decoder. For this language model, we included even singleton events, and applied a modified Knesser-Ney smoothing. We first investigated using a 7-gram language model. While early tests employing this language

model reduced the net word error rate, latter tests using a more mature acoustic model gave about the same results for the 7-gram and 3-gram language models. For the final system we conservatively chose to use the 3-gram for rescoring [10].

Overall Recognition Structure

The recognition system was composed of the following stages:

1. Segmentation, classification, and clustering
2. Initial-pass recognition
3. Initial-pass best-path search
4. Acoustic adaptation
5. Second-pass recognition
6. Second-pass best-path search
7. N-best rescoring

Segmentation and Preprocessing

In this evaluation, we were faced with two different evaluation scenarios. In the first, 'partitioned evaluation' (PE), we were given a set of shows, and a list of break-points corresponding to places where the speaker changed, the acoustic condition changed (*e.g.* background music started or stopped), a major linguistic boundary occurred (*e.g.* at a change in topic), and the like. In the second, 'unpartitioned' scenario (UE), we were given whole or near-whole broadcasts, and were faced with the task of separating these very long segments into pieces short enough to be suitable as input for our recognizer.

We were also concerned that even in the partitioned evaluation scenario, we might be confronted with segments of several minutes in length, which would overwhelm the capabilities of the Sphinx-3 decoder. As additional incentive to trim long segments into shorter ones, we reasoned that in order to get approximately the same coverage of points of uncertainty, the number of N-best hypotheses needed should be exponential in the length of the input.

With this all in mind, we decided to attack the unpartitioned evaluation problem strongly, noting that these problems with the partitioned evaluation would be taken care of as well. In the end we used the same tools in somewhat different order for the PE and UE. These are described briefly below; a more thorough treatment can be found in [11].

For the partitioned evaluation, the initial pass consisted of classifying each NIST-supplied segment, clustering segments by similarity, and splitting long segments into sub-segments.

For the unpartitioned evaluation, the initial pass consisted of automatic segmentation, classification, and clustering. The pieces used were the same as for the PE, though the order of application differed.

Classification: Each segment in the test set was identified as either "Full Bandwidth" (FBW) or "Half Bandwidth" (HBW) using mixture Gaussian models. The FBW model contained mixtures of 16 Gaussian densities and was trained using acoustic data provided by LDC that mapped to either the F0 or F1 focus condition. The HBW model contained 8 densities and was trained using hand-labeled telephone segments from the 1995 Hub-4 training data.

We found that it was important to classify each segment mechanically. Obviously in the UE we would not have any hints. Even in the PE, and in the training data, not all segments marked as 'F2' were, in fact, speech over a telephone. We found that even for the 'F2' subset, it was better to use the choice of our classifier.

Our classifier also chose between male-only, female-only, and both-gendered full-bandwidth models. As mentioned above on the development test set we found no advantage to separating the data for gender-specific models. So in the end we mapped our four-way model choice into the two (FBW and HBW) that we actually used.

Clustering: Segments were clustered using an acoustical similarity metric similar to that used by Hwang for comparing statistical distributions [1]. First, single mixture Gaussian parameters for each utterance were estimated using maximum likelihood techniques. Then, the models estimated from each segment were clustered together if the symmetric cross-entropy between them was smaller than an empirically-derived threshold. We insured that after the clustering operation, each cluster contained at least 10 seconds of data.

Sub-segmentation: To reduce the length of the decoded utterances to 30 seconds, silences in each utterance were located, and the utterances broken at that point. A silence was located at frame x when the following criteria were met (1 frame equals 10 ms):

1. The average power of the interval $[x-7, x+7]$ was more than 8 dB lower than the power of the interval $[x-200, x+200]$.
2. The range of the power of the interval $[x-7, x+7]$ was less than 10 dB.

Automatic Segmentation and Silence Detection: Initially, the long audio streams were chopped into smaller segments, at points determined to be acoustic boundaries. These acoustic boundaries were found using a Cross Entropy similarity metric [11], where the statistics of 250 frames (2.5 sec) of data to the left and right of the boundary were compared. When the similarity was at its local minimum, and was also smaller than a predefined threshold, an acoustic boundary was found.

Silences near the boundaries were then located as above, and the utterances broken at that point.

Initial-pass recognition: The initial recognition was done with a straight-forward continuous- density Viterbi beam search, using the models described above. In addition to a hypothesis containing words and their times, this recognition produced a word lattice for each sub-segment.

These lattices were then searched for the global best path according to the trigram grammar. Briefly, the lattice was converted into a directed acyclic graph, with nodes corresponding to words that started at a particular time, and weighted arcs indicating which words could follow which other words, weighted by a combination of the acous-

tic score for a word and the language model score implied by the transition. The only acoustic scores used in this search were retrieved from the lattice, but the language model scores were recomputed. As a result, this part was much quicker than the search that produced the lattice. The result of this search was a globally optimal hypothesis with respect to higher-span (3-or-more-gram) language models [5, 6].

Acoustic adaptation: The HMM means were adapted using Maximum Likelihood Linear Regression (MLLR) [4] as described above. This adaptation was performed with a single regression matrix, based on the best theory produced by the best-path lattice search as above (or, if that failed due to resource limitations, the Viterbi search result).

The subsegments were adapted into groups, according to the clustering derived in the initial segmentation and clustering step. Full- and Half-bandwidth subsegments were not clustered together.

Second-pass recognition: Each subsegment was then decoded again, using the acoustic models adapted in the previous step. Again a lattice was produced for each subsegment. This lattice was both searched for the global best path, as above. An N-best search [9] over the lattice was also done at this point. The Viterbi and best-path results and the N-best lists were passed on to the rescoring step. For the evaluation system we used 200-best lists.

N-best rescoring: The N-best lists generated using the supplemented vocabulary [10] were processed to convert the phrases and acronyms into their constituent words and letters, respectively. N-best rescoring was then performed in the space of the unsupplemented vocabulary.

The Viterbi search, best-path, and N-best hypotheses were rescored as described above, using the Kneser-Ney smoothed trigram language model. The highest scoring hypothesis according to this rescoring was output.

Reformatting and collating: Finally, the hypotheses for all segments were collected together, reformatted to be suitable as input to the NIST scoring tools, and sorted into original time order.[1]

Overall System Architecture

Two of the greatest challenges in putting together a complex system are managing the flow of information, and detecting failures in order to correct them when they occur.

We first envisioned a quite complex system, with many separate pieces of information flowing from module to module. We quickly came to the conclusion that handling all of this data in such a fashion, and ensuring its consistency and correctness, would have been a nightmare. We chose to circumvent these problems by changing the format of the control files that our system used, and packing all the segment-dependent information into the "name" of the subsegment.

[1]This step of processing, as well as many others implied above, was made practically uninteresting by use of the Perl programming language [12].

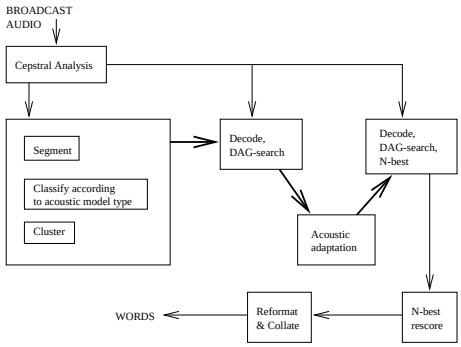

Figure 1: Decoder Architecture

Each "name" consisted of:

1. File name
2. Sub-segment unique serial number
3. Focus condition, or "UE"
4. Starting frame
5. Ending frame
6. Story ID, or "0" for UE
7. Acoustic model to use (from classifier): M, F, G, or P
8. Acoustic cluster number
9. NIST segment ID, or "0000" for UE

For example, for the first subsegment of the PE:

```
file1_0000_F0_000062_000766_001_M_0001_0001
```

By overloading the 'name' in this fashion, the actual system complexity was considerably reduced, resulting in the fairly clean architecture shown in figure 1.

As an added benefit, this allowed us to easily check for errors. For example, if the above subsegment was missing from the output of the second pass decode, we would know to look at the decode run for the 'M' models applied to cluster number '0001'.

Experimental results

Our evaluation results are shown in tables 1 and 2, as well as a breakdown by F-condition of some intermediate steps in the processing of the evaluation runs.

Murphy's Law[2] is a specter that looms over any evaluation. This year, we were stung by a subtle bug that fortunately had only a small effect on our reported performance. The Sphinx-3 decoder occasionally exhibited a fault that caused the end-of-segment token `</s>` to consume a large fraction of the end of a segment. Though we thought we had solved this through a judicious choice in model configuration, this detail did not make it into the final configuration. And, of course, the development test run with which we validated the final configuration, did not expose the bug.

[2]"Anything that can go wrong will go wrong." [2]

	Overall		*F0*	*F1*	*F2*	*F3*	*F4*	*F5*	*FX*
1st pass Viterbi	38.8		28.8	33.8	44.9	45.1	45.6	45.2	64.6
1st pass best-path	37.3	(+4%)	27.2	32.4	43.2	43.3	45.7	45.8	61.8
2nd pass best-path	35.5	(+9.3%)	26.1	32.3	39.7	37.3	43.9	38.1	57.8
Rescored (final)	34.9	(+11%)	25.8	32.1	38.6	36.6	43.7	36.5	55.8

Table 1: Partitioned Evaluation

	Overall		*F0*	*F1*	*F2*	*F3*	*F4*	*F5*	*FX*
1st pass Viterbi	39.1		27.2	34.4	45.8	50.0	45.6	40.8	66.7
1st pass best-path	37.8	(+3.4%)	26.0	33.5	44.7	48.4	45.0	40.8	62.9
2nd pass best-path	36.5	(+7.1%)	24.8	33.7	39.8	48.8	42.5	38.8	60.3
Rescored (final)	35.9	(+8.9%)	24.7	33.1	39.1	48.4	42.1	35.5	58.3

Table 2: Unpartitioned Evaluation

As a result, we lost approximately 120 words of output in the partitioned evaluation. By fixing the bug and re-running the affected segments, we now estimate that this bug contributed 0.4% (absolute) to our overall word error rate, and 1.3% to our F0 word error rate. Interestingly, our unpartitioned evaluation was not effected by this bug.

A comparison of PE and UE results shows how well our efforts in automatic segmentation paid off. Our overall scores show a 3% relative difference between the PE and UE (4% post-bug-fix). This demonstrates the effectiveness of our new segmentation system.

Like some other sites, we were somewhat surprised by the discrepancy between our development and evaluation test results. We observed a much higher error rate on the evaluation data compared to the development test. So far we have not been able to adequately characterize this effect.

Acknowledgements

This research was sponsored by the Department of the Navy, Naval Research Laboratory under Grant No. N00014-93-1-2005.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

We thank Mei-Yuh Hwang and the Microsoft Research Speech Group for their valuable suggestions and contributions.

We also thank Raj Reddy, Lin Chase, and the rest of the speech group, and Mary Placeway, for contributions to this work.

REFERENCES

1. Mei-Yuh Hwang. Subphonetic Acoustic Modeling for Speaker-Independent Continuous Speech Recognition. PhD Thesis CMU-CS-93-230, Carnegie Mellon University, 1993.
2. ed. Jim Russell. *Murphy's law ...* Celestial Arts, 1978.
3. F. Kubala, T. Anastasakos, H. Jin, L. Nguyen, and R. Schwartz. Transcribing Radio News. In *Proc. ICSLP*. University of Delaware and A.I. duPont Inst., 1996.
4. C. J. Leggetter and P. C. Woodland. Speaker Adaptation of HMMs using Linear Regression. F-INFENG Tech Report 181, Cambridge University Engg. Dept., June 1994.
5. Mosur Ravishankar. Efficient Algorithms for Speech Recognition. PhD Thesis CMU-CS-96-143, Carnegie Mellon University, 1996.
6. Mosur Ravishankar. Some Results on Search Complexity vs. Accuracy. In *Proc. Spoken Language Systems Technology Workshop*. Morgan Kaufmann Publishers, 1997.
7. V. Parikh, B. Raj, and R. Stern. Experiments on Compensation and Adaptation. In *Proc. Spoken Language Systems Technology Workshop*. Morgan Kaufmann Publishers, 1997.
8. W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C*. Cambridge University Press, second edition, 1992.
9. R. Schwartz, *et. al.* New uses for the N-best Sentence Hypotheses within the BYBLOS speech recognition system. In *Proc. ICASSP-92*, volume I, pages 1 – 4, 1992.
10. K. Seymore, S. Chen, M. Eskenazi, and R. Rosenfeld. Language and Pronunciation Modeling in the CMU 1996 Hub 4 Evaluation. In *Proc. Spoken Language Systems Technology Workshop*. Morgan Kaufmann Publishers, 1997.
11. M. Siegler, U. Jain, B. Raj, and R. Stern. Automatic Segmentation, Classification, and Clustering of Broadcast News Audio. In *Proc. Spoken Language Systems Technology Workshop*. Morgan Kaufmann Publishers, 1997.
12. Larry Wall and Randal L. Schwartz. *Programming perl*. O'Reilly & Associates, 1990.